Math 142: Modeling **Name:**_____

Homework # 5 **Due: Friday Nov 3**

Directions Complete the exercises. Your solutions to the exercises should be submitted to Gradescope before the indicated due date above. Please follow rules regarding Gradescope submission as described in the syllabus.

References Except for the help of the instructor or TAs and the class textbooks and notes, if you use any resources, for example, a book, a website, or you discussed with your friends, please acknowledge them in this References section.

- I discussed Problem ?? with STUDENT A, STUDENT B, ...
- I used BOOK/WEBSITE to help me do Problem ??.

Exercises

1. **Population genetics** We have argued that real populations have carrying capacities, at which rates of birth and death are equal. We can study the diversity of such populations (how many different cells will be present), using a Moran process model. Consider a population of N cells, which we initially label $1, 2, 3, \ldots$. You task is to develop a simulation of how many descendants each of these starting cells has during the following process.

   In each unit of time, imagine that a randomly chosen cell among the N divides. This produces another cell with the same label (so if a cell with label 2 divides, then the new cell also has label 2). But to keep the number of cells constant, a cell must die. Take another randomly chosen cell among the N and remove it from the population. (It is possible that the same cell will divide and die in a single time step - that's ok). e.g. Suppose we start with 6 cells: $1, 2, 3, \ldots 6$ In our first step we find that cell 5 divides, and cell 2 dies. After the first step our cells are now labeled: $1, 3, 4, 5, 5, 6$.

   (a) Simulate 1000 time steps of this process using 6 cells. Show that after enough time steps all of the cells eventually have the same label (i.e. descend from only one of your starting cells).

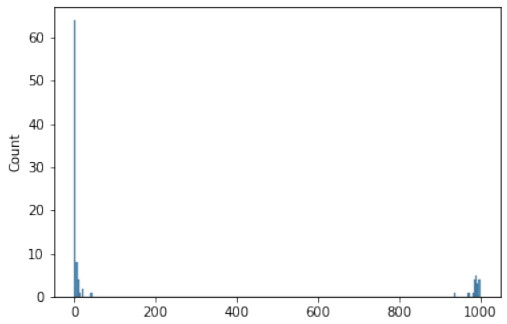
   (b) Simulate 100 replicate populations, each with $N = 6$ cells. Show that they all, eventually each replicate population becomes homogeneous.

```
: def all_equal(arr):
      if np.sum(arr==arr[0])==len(arr):
          return True
      else:
          return False

: def diversity_sim(N,sim_number,steps):
      descendants=np.arange(1,N+1)
      simulations=np.array([descendants]*sim_number)
      for step in np.arange(steps+1):
          for sim in simulations:
              x_b=min(math.floor(random.uniform(0,1)*N),N-1)
              x_m=min(math.floor(random.uniform(0,1)*N),N-1)
              sim[x_m]=sim[x_b]
      return all_equal(np.array([all_equal(sim) for sim in simulations]))

: diversity_sim(6,1,1000)

: True

: diversity_sim(6,100,1000)

: True
```

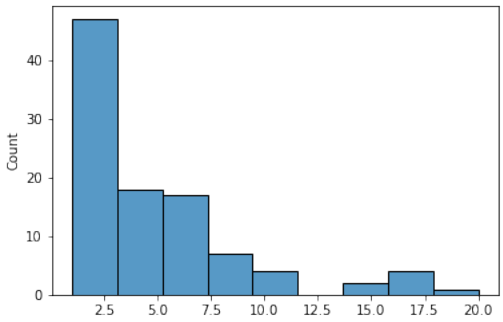
(This answers both (a) and (b))

(c) Explain: After 1000 time steps, what is the average number of descendants that cell number 1 should have among the 6 cells? Argue mathematically (hint: every cell should have the same average number of descendants), and then compare with the answer obtained from your numerical simulation.

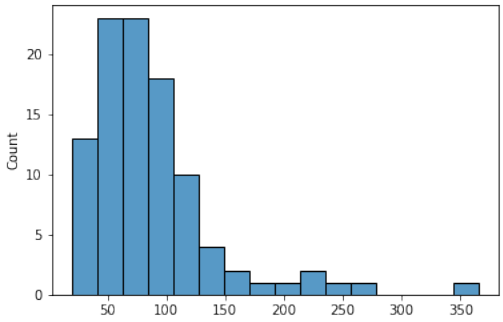
If, on average, every cell should have the same number of descendants, we should expect each cell to produce $\frac{1000}{6}$ descendants. Using numerical simulations, we obtain an average of 199.43 with distribution:



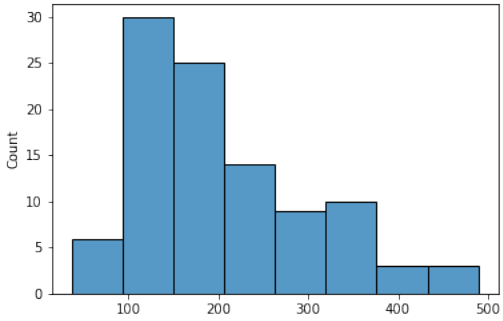
(d) The same homogenization occurs in populations of different sizes. Measure the average time taken for the population to become homogeneous in $N = 3, N = 10, N = 15$ cells.



N=3 mean=5.02

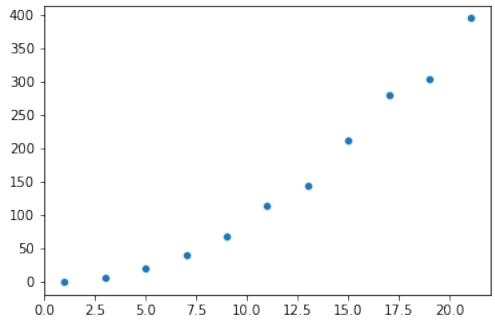


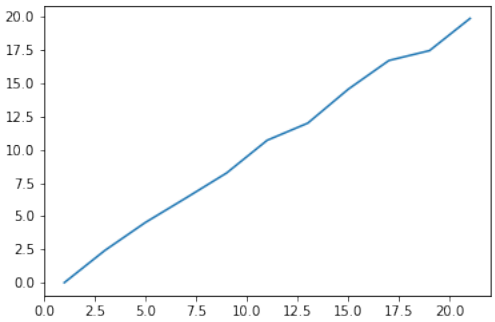
N=10 mean 88.1



N=15 mean 205.58

(e) It has been argued that the average number of steps for the population to homogenize is proportional to N^2. Plot your data from (d) in a way that tests this relationship. The process of homogenization is known as genetic drift, and is an important force in evolution. Although our model is for cells dividing, it also occurs in populations that reproduce sexually. In particular we inherit our mitochondrial DNA is inherited directly from our mothers. Through mitochrondial DNA it can be shown that we are each descendants of one of between 7 and 29 women (different genetic mapping methods produce different answers).





The scatter plot shows that there appears to be some evidence for a quadratic relationship. The relationship between the square roots of the average time it takes to reach homogenization vs. number of cells appears to be somewhat linear.

2. In class we studied the logistic model for the density dependent growth of populations:

$$\frac{dN}{dt} = r\left(1 - \frac{N}{K}\right)N, \quad N(0) = N_0,$$

and we derived the solution:

$$N(t) = \frac{K}{1 - \frac{N_0 - K}{N_0}e^{-rt}},$$

and showed explicitly that $N(t) \to K$ as as $t \to \infty$, so long as $N_0 > 0$.

(a) Check explicitly that $N(0) = N_0$.
$N(0) = \frac{K}{1-\frac{N_0-K}{N_0}e^{-r\cdot 0}} = \frac{K}{\frac{N_0}{N_0}-\frac{N_0-K}{N_0}\cdot 1} = \frac{K}{\frac{K}{N_0}} = N_0$

(b) Suppose that $N_0 < K$. Show by computing $N'(t)$, that $N(t)$ is an increasing function.
$N'(t) = (K \cdot (1 - \frac{N_0-K}{N_0}e^{-rt})^{-1})' = K(-1)\cdot(1 - \frac{N_0-K}{N_0}e^{-rt})^{-2} \cdot r\frac{N_0-K}{N_0}e^{-rt}$
$\frac{N_0-K}{N_0} < 0$ because $N_0 < K$, so let $P = -\frac{N_0-K}{N_0}$ be a positive number.
Thus, $N'(t) = \frac{KPe^{-rt}}{(1+Pe^{-rt})^2} > 0$ because $K, P, e^{-rt} > 0$ for all t and dividing a positive number by a positive number yields a positive number. Since $N'(t) \geq 0$ for all t, $N(t)$ is an increasing function.

(c) Suppose that $N_0 > K$. By computing $N'(t)$, determine whether $N(t)$ is an increasing or decreasing function. $0 < \frac{N_0-K}{N_0} < 1$ because $N_0 > K$, so let $P = \frac{N_0-K}{N_0}$ be a positive number.
$N'(t) = \frac{-KP}{(1-Pe^{-rt})^2} < 0$ because $0 < 1 - Pe^{-rt}$ $(0 < P, e^{-rt} < 1 \Rightarrow 0 < P \cdot e^{-rt} < 1)$ and $-KP < 0$.
Since $N'(t) \leq 0$ for all t, $N(t)$ is an decreasing function.

(d) Suppose that $N_0 = K$. What is the behavior of the solution?
$\frac{N_0 - K}{N_0} = 0$ because $N_0 = K$, so $N'(t) = 0$. Thus, $N(t) = K$ is a constant function

3. Draw phase portraits and possible solution curves for the following autonomous differential equations:

(a)
$$\frac{dx}{dt} = x^3 - 4x.$$

(b)
$$\frac{dy}{dx} = ye^{-y}$$

(c)
$$\frac{dy}{dt} = \frac{y-2}{y^2-1} \quad , \quad y \neq \pm 1$$

(d)
$$\frac{dx}{dt} = \sin 2x \quad , \quad 0 \leq x \leq 2\pi$$

4. Consider the ODE $\frac{dy}{dt} = g(y)$. The graphs below are the graphs of $g(y)$. For each graph , draw the possible solution curves $y(t)$ as a function of t. Your sketch of the solution should show where the solution converges to (if it converges), and also where the solution curve is concave up, and also where it is concave down.

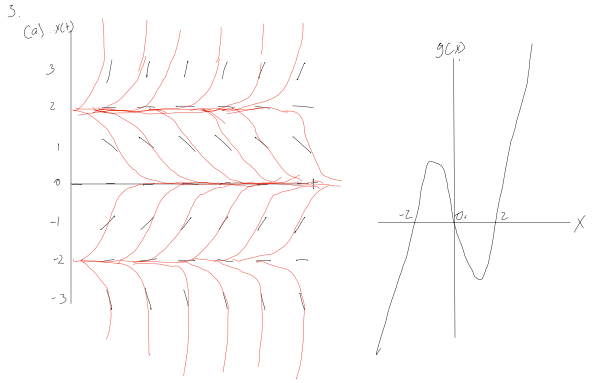
5. Look at the sets of solution curves shown in the graphs below. For each part, draw as well as you can, the corresponding phase portrait for the differential equation (that is, plot $g(y)$, the function on the right hand side of the differential equation $\left(\frac{dy}{dt} = g(y)\right)$. _____

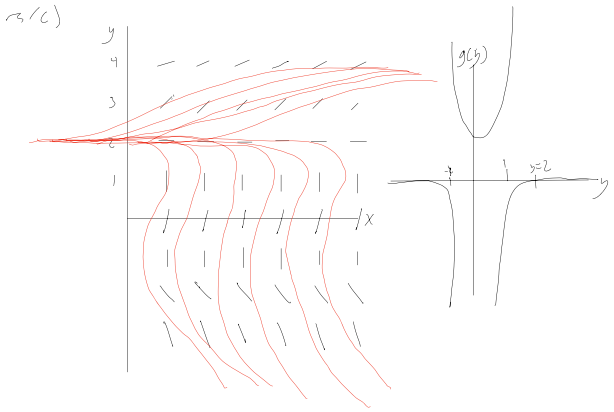
6. By separately graphing the two terms on the right hand side of each of the differential equations below, determine how many fixed points each equation has, and find their stability. (It is not necessary to find the numerical locations of any fixed points):

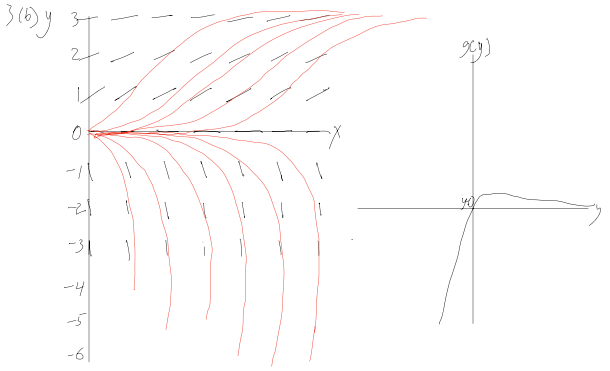
(a)
$$\frac{dx}{dt} = e^{-x^2} - x,$$

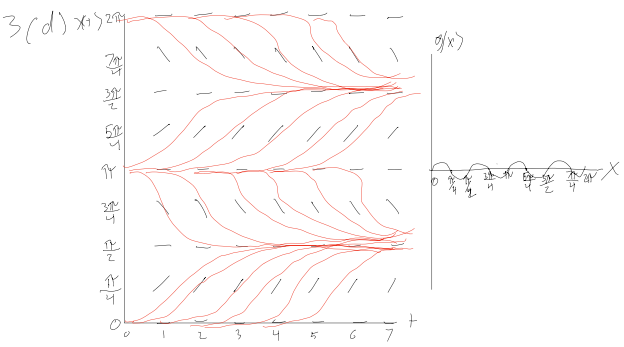
(b)
$$\frac{dy}{dt} = y^4 - (1+y),$$

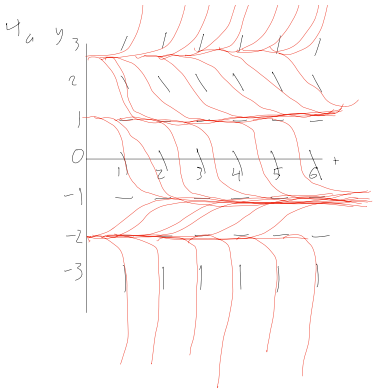
(c)
$$\frac{dy}{dx} = y^3 - e^{-y}.$$

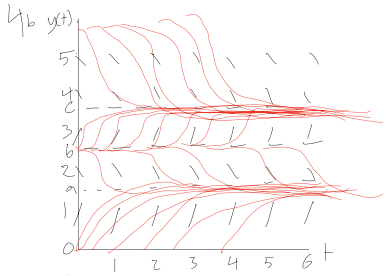

3.
(a) x(t)
g(x)
X

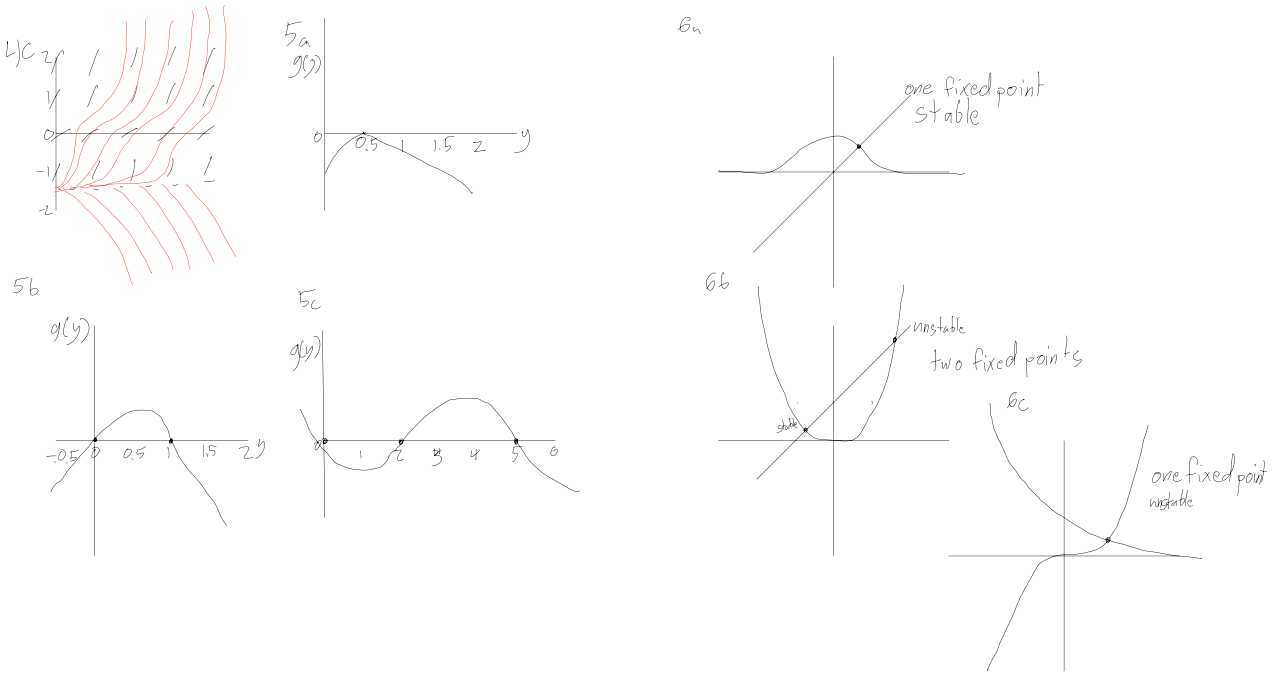

3(c)
y
g(y)
y=2


3(b) y
g(y)


3(d) x(t)
g(x)


4a y


4b y(t)



7. Submit the code you used for any and all of the problems. (Print pdf the code) Either lump it all together at the end or when matching problems on Gradescope, select all pages of pdf that has code if you included code within the solution to each answer.

In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import random
import math
```

In [2]:
```
def all_equal(arr):
    if np.sum(arr==arr[0])==len(arr):
        return True
    else:
        return False
```

In [3]:
```
def diversity_sim(N,sim_number,steps):
    descendants=np.arange(1,N+1)
    simulations=np.array([descendants]*sim_number)
    for step in np.arange(steps+1):
        for sim in simulations:
            x_b=min(math.floor(random.uniform(0,1)*N),N-1)
            x_m=min(math.floor(random.uniform(0,1)*N),N-1)
            sim[x_m]=sim[x_b]
    return all_equal(np.array([all_equal(sim) for sim in simulations]))
```

In [4]:
```
diversity_sim(6,1,1000)
```

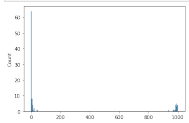
Out[4]:
```
True
```

In [5]:
```
diversity_sim(6,100,1000)
```

Out[5]:
```
True
```

In [6]:
```
def diversity_sim_descendant_count(N,sim_number,steps,num_count):
    descendants=np.arange(1,N+1)
    simulations=np.array([descendants]*sim_number)
    counter=np.zeros(sim_number)
    for step in np.arange(steps+1):
        for index,sim in enumerate(simulations):
            x_b=min(math.floor(random.uniform(0,1)*N),N-1)
            x_m=min(math.floor(random.uniform(0,1)*N),N-1)
            sim[x_m]=sim[x_b]
            if sim[x_b] == num_count:
                counter[index]+=1
    sns.histplot(counter)
    return np.percentile(counter,[0,25,50,75,100]),np.mean(counter)
```

In [14]:
```
n_1=diversity_sim_descendant_count(6,100,1000,1)
plt.savefig("div_sim_descendant_count")
```

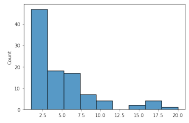


In [15]:
```
n_1
```

Out[15]:
```
(array([  0.  ,   0.  ,   1.  ,   9.25, 999.  ]), 199.43)
```

In [8]:
```
def diversity_sim_homogenization_rate(N,sim_number,steps,graph=True):
    descendants=np.arange(1,N+1)
    completed = np.array([np.empty(2)]*sim_number)
    simulations=np.array([descendants]*sim_number)
    for step in np.arange(steps+1):
        if np.equal(completed.transpose()[0],np.arange(sim_number)).all()==True:
            break
        for index,sim in enumerate(simulations):
            if np.isin(index,completed.transpose()[0])==True:
                continue
            else:
                x_b=min(math.floor(random.uniform(0,1)*N),N-1)
                x_m=min(math.floor(random.uniform(0,1)*N),N-1)
                sim[x_m]=sim[x_b]
                if all_equal(sim)==True:
                    completed[index][0]=index
                    completed[index][1]=step
    if graph==True:
        sns.histplot(completed.transpose()[1])
    return np.percentile(completed.transpose()[1],[0,25,50,75,100]),np.mean(completed.transpose()[1])
```

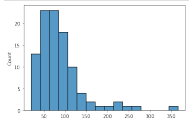
In [16]:
```
n_2=diversity_sim_homogenization_rate(3,100,1000)
plt.savefig("div_sim_hom_3")
```



In [17]:
```
n_2
```

Out[17]:
```
(array([ 1.,  2.,  4.,  7., 20.]), 5.02)
```

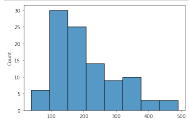
In [18]:
```
n_3=diversity_sim_homogenization_rate(10,100,1000)
plt.savefig("div_sim_hom_10")
```



In [19]:
```
n_3
```

Out[19]:
```
(array([ 20.,  54.,  74., 106., 365.]), 88.1)
```

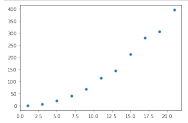
In [20]:
```
n_4=diversity_sim_homogenization_rate(15,100,1000)
plt.savefig("div_sim_hom_15")
```



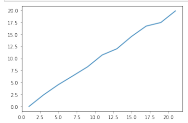
In [21]:
```
n_4
```

Out[21]:
```
(array([ 38.  , 124.75, 184.5 , 261.75, 489.  ]), 205.58)
```

In [12]:
```
data=np.array([diversity_sim_homogenization_rate(N,100,1000,False)[1] for N in np.arange(1,23,2)])
sns.scatterplot(x=np.arange(1,23,2),y=data)
plt.savefig('div_sim_hom_rates')
```



In [13]:
```
sns.lineplot(x=np.arange(1,23,2),y=[math.sqrt(d) for d in data])
plt.savefig('div_sim_hom_regression')
```



In [ ]:
```
```